**Stanut Denisa - Alexandra**
**Grupa 30224**

# MIPS32 Pipeline

## Elemente functionale

- Toate elementele sunt functionale (Unitatea de executie, Instruction fetch, Unitatea de memorie, Unitatea de decodificare a instructiunilor, Afisorul SSD si Unitatea de control)

## Elemente nefunctionale

- Nu e cazul.

## Probleme intampinate

- Nu am intampinat probleme la implementare.

## Testare

- Programul a fost testat atat pe placa in cadrul laboratorului cat si in simulator.

## Instructiuni alese suplimentar

**SRA (Shift-Right Arithmetic)**

- **Descriere:** Deplasare aritmetica la dreapta pentru un registru, rezultatul fiind memorat in alt registru, repetandu-se valoarea bitului de semn.
- **RTL:** $\$d \leftarrow \$t >> h;\ PC \leftarrow PC + 4$
- **Sintaxa:** `sra $d, $t, h`
- **Format:** `000000 00000 ttttt ddddd hhhhh 000011`
- **Semnale control:** `RegDst←1, ExtOp←X, ALUSrc←0, Branch←0, Brgez←0, Jump←0, MemWrite←0, MemtoReg←0, RegWrite←0, AluOp←10, function←000011, AluCtrl←110(>>a)`

**SLLV (Shift-Left Logical Variable)**

- **Descriere:** Deplasare logica la stanga pentru un registru, cu un numar de pozitii indicat de alt registru, rezultatul fiind memorat intr-un al treilea registru.
- **RTL:** $\$d \leftarrow \$t << \$s;\ PC \leftarrow PC + 4$
- **Sintaxa:** `sllv $d, $t, $s`
- **Format:** `000000 sssss ttttt ddddd 00000 000100`
- **Semnale control:** `RegDst←1, ExtOp←X, ALUSrc←0, Branch←0, Brgez←0, Jump←0, MemWrite←0, MemtoReg←0, RegWrite←0, AluOp←10, function←000100, AluCtrl←111(<<lv)`

**ANDI (AND Immediate)**

- **Descriere:** SI logic intre un registru si o valoare imediata, rezultatul ajunge in alt registru.
- **RTL:** $t ← $s & ZE(imm); PC ← PC + 4
- **Sintaxa:** `andi $t, $s, imm`
- **Format:** `001100 sssss ttttt iiiiiiiiiiiiiiii`
- **Semnale control:** `RegDst←0, ExtOp←0, ALUSrc←1, Branch←0, Brgez←0, Jump←0, MemWrite←0, MemtoReg←0, RegWrite←0, AluOp←11, function←XXXXXX, AluCtrl←001`

**BGEZ (Branch on Greater than or Equal to Zero)**

- **Descriere:** Se executa un salt conditionat daca un registru este mai mare sau egal cu 0.
- **RTL:** `if $s ≥ 0 then PC ← (PC + 4) + (SE(offset) << 2) else PC ← PC + 4`
- **Sintaxa:** `bgez $s, offset`
- **Format:** `000001 sssss 00000 oooooooooooooooo`
- **Semnale control:** `RegDst←X, ExtOp←1, ALUSrc←0, Branch←0, Brgez←1, Jump←0, MemWrite←0, MemtoReg←X, RegWrite←0, AluOp←01, function←XXXXXX, AluCtrl←100`

# Modificari particulare pentru implementarea instructiunilor alese

- Am adaugat pentru toate instructiunile alese modificarea semnalelor de control in UC, respectiv in EX, insa pentru instructiunea BGEZ (Branch on greater than or equal to zero) am adus mai multe modificari in cadrul proiectului. Pentru aceasta am adaugat in plus un flag in componenta EX care e 1 daca cel mai semnificativ bit al rezultatului e egal cu 0, adica daca numarul e mai mare sau egal cu 0. De asemenea am mai adaugat si in top level o modificare care consta in rezultatul semnalului PCSrc care ia valoarea:
`(Zero_EX_MEM and Branch_EX_MEM) or (Gez_EX_MEM and Br_gez_EX_MEM)`